Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▸ BURBAN
Nom d'usage ▸
Prénom ▸ Yann
Adresse ▸ Entrez votre adresse ici.

Titre professionnel visé

Développeur (euse) Web et Web Mobile

MODALITE D'ACCES :

☒ Parcours de formation
☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen**.

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

http://travail-emploi.gouv.fr/titres-professionnels

Sommaire

Exemples de pratique professionnelle

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°1 ▸ Développement de l'interface de filtrage et d'affichage dynamique des articles pour une application d'enchères

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du projet Enchere, réalisé en équipe durant ma formation à l'ENI, j'ai conçu et développé l'interface utilisateur (front-end) dédiée à la recherche et au filtrage des articles mis en vente sur la plateforme. Cette application permet aux utilisateurs inscrits de consulter, filtrer et enchérir sur des articles proposés par d'autres membres.

Concrètement, j'ai mis en place :

- Une gestion dynamique de l'en-tête de l'application :
  - L'en-tête s'adapte selon l'état de connexion de l'utilisateur (liens d'inscription/connexion ou informations personnelles et accès aux fonctionnalités réservées).
- Une section de filtres dynamiques sur la page de consultation des articles, permettant de restreindre la liste selon plusieurs critères (catégorie, prix, état de la vente, etc.).
- L'utilisation de Thymeleaf comme moteur de templates pour générer dynamiquement les pages HTML côté serveur. Grâce à Thymeleaf, les données (utilisateur, filtres, articles) sont injectées directement dans les vues, ce qui permet d'afficher ou masquer certains éléments selon le contexte (connexion, rôle, etc.).
- Une interface responsive, adaptée à tous types d'écrans.

Tâches réalisées :

- Création des templates HTML avec Thymeleaf pour l'en-tête, les filtres et la liste des articles.
- Utilisation des balises et expressions Thymeleaf pour afficher dynamiquement les données et gérer l'affichage conditionnel (ex : th:if, th:each).
- Mise en place des contrôleurs Spring Boot pour gérer les requêtes :
  - GET : récupération et affichage des articles et des filtres (ex : /articles).
  - POST : traitement des formulaires de filtrage ou d'enchère, puis redirection ou affichage des résultats filtrés.
- Liaison entre le front-end et le back-end via les modèles Spring, facilitant le passage des données entre le contrôleur et la vue.

Ce choix d'architecture permet d'offrir une interface dynamique, sécurisée et adaptée au contexte de chaque utilisateur, tout en facilitant la maintenance et l'évolution du projet.

2. Précisez les moyens utilisés :

- HTML / CSS / Tailwindcss : structuration et mise en forme des pages.
- JavaScript : interactions utilisateur complémentaires.
- Thymeleaf : génération dynamique des pages HTML et gestion de l'affichage conditionnel.
- Spring Boot : gestion des routes, des contrôleurs et du traitement des requêtes GET/POST.

3. Avec qui avez-vous travaillé ?

Nous avons mené ce projet à trois, en présentiel à l'école ENI à Nantes, avec un formateur référent jouant le rôle du client. La diversité de nos niveaux d'expérience a favorisé une bonne dynamique d'équipe et une collaboration efficace. En début de journée, nous réalisions un daily (entrevu concise) pour revoir les points d'avancement ou de blocage de chacun, selon une méthode inspirée de l'agilité.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***ENI École Informatique – Nantes***

Chantier, atelier, service ▸ **Projet de formation Développeur Web et Web Mobile**

Période d'exercice ▸ **Du 10/02/2025 au 21/02/2025**

5. Informations complémentaires *(facultatif)*

Les annexes 1 à 4 sont les visuels et les codes permettant l'affichage de la page d'accueil du site Enchères.

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n°1 ▸ Développement des requêtes d'accès aux données du projet Enchères

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du projet d'application web d'enchères développé en équipe à l'ENI, j'ai été responsable de la mise en place de la fonctionnalité d'affichage des articles en vente.
Cette fonctionnalité permet à tout utilisateur de consulter la liste des articles disponibles, avec possibilité de filtrer par nom ou catégorie si aucun utilisateur n'est connecté.

Lorsque l'utilisateur est connecté, des filtres supplémentaires sont disponibles, comme la sélection des articles selon leur statut (enchères en cours, remportées, mises en vente, etc.) ou selon l'option d'achat ou de vente, offrant ainsi une expérience de recherche plus personnalisée et adaptée à son profil.

Tâches effectuées :

- Création de l'interface ArticleVenduDAO pour définir les méthodes d'accès aux articles (recherche, filtrage, récupération par ID, etc.).
- Implémentation de l'interface dans la classe ArticleVenduDAOImpl : écriture des requêtes SQL paramétrées pour interroger la base de données et retourner les articles selon les critères demandés, tout en limitant l'exposition des données sensibles et en prévenant les injections SQL.
- Définition de l'interface ArticleVenduService pour exposer les opérations métier liées aux articles (recherche, filtrage, etc.).
- Implémentation de la couche service dans ArticleVenduServiceImpl : appel aux méthodes DAO, gestion de la logique métier (vérification des filtres, gestion des exceptions métier pour éviter la fuite d'informations techniques), et vérification des droits de l'utilisateur avant d'exécuter certaines opérations sensibles (modification, suppression).

Ces mesures garantissent que seules les personnes autorisées peuvent accéder ou modifier les données et que le code reste protégé contre les attaques courantes (injection, élévation de privilèges, fuite d'informations).

La structuration du code, la séparation des responsabilités et l'utilisation des bonnes pratiques (DAO, Service, Contrôleur, sécurité) assurent la robustesse et la maintenabilité de l'application.

2. Précisez les moyens utilisés :

Environnement de développement : Eclipse

Langage : Java

Frameworks : Spring Boot (MVC, Security), Thymeleaf

Base de données : SQL Server

Outils : Gradle, GitHub

Méthodologie : Architecture MVC avec séparation claire des couches (DAO, Service, Contrôleur)

Sécurité :

Les données sensibles (ex : identifiants utilisateur) ne sont jamais exposées dans les réponses.

Les requêtes SQL sont paramétrées pour éviter les injections.

3. Avec qui avez-vous travaillé ?

Nous avons mené ce projet à trois, en présentiel à l'école ENI à Nantes, avec un formateur référent jouant le rôle du client. La diversité de nos niveaux d'expérience a favorisé une bonne dynamique d'équipe et une collaboration efficace. En début de journée, nous réalisions un daily (entrevu concise) pour revoir les points d'avancement ou de blocage de chacun, selon une méthode inspirée de l'agilité.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***ENI École Informatique – Nantes***

Chantier, atelier, service ▸ **Projet de formation Développeur Web et Web Mobile**

Période d'exercice ▸ **Du 10/02/2025 au 21/02/2025**

5. Informations complémentaires *(facultatif)*

Les annexes 5 à 10 sont les interfaces, les class et le diagramme de classe permettant de récupérer et de retourner au client les articles en fonction des filtres sélectionnés et de son statut sur le site Enchères.

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Développeur Web et Web Mobile	**ENI Nantes**	

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] *BURBAN Yann*,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur(e) des réalisations jointes.

Fait à *Saint Léger les Vignes* le 25/06/2025

pour faire valoir ce que de droit.

Signature :

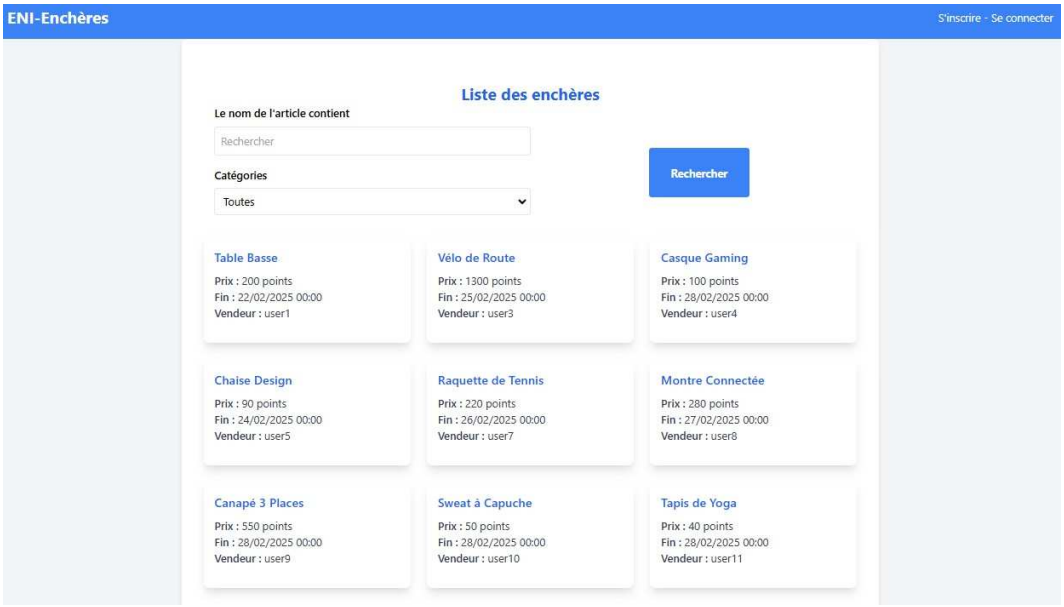
Documents illustrant la pratique professionnelle

(facultatif)

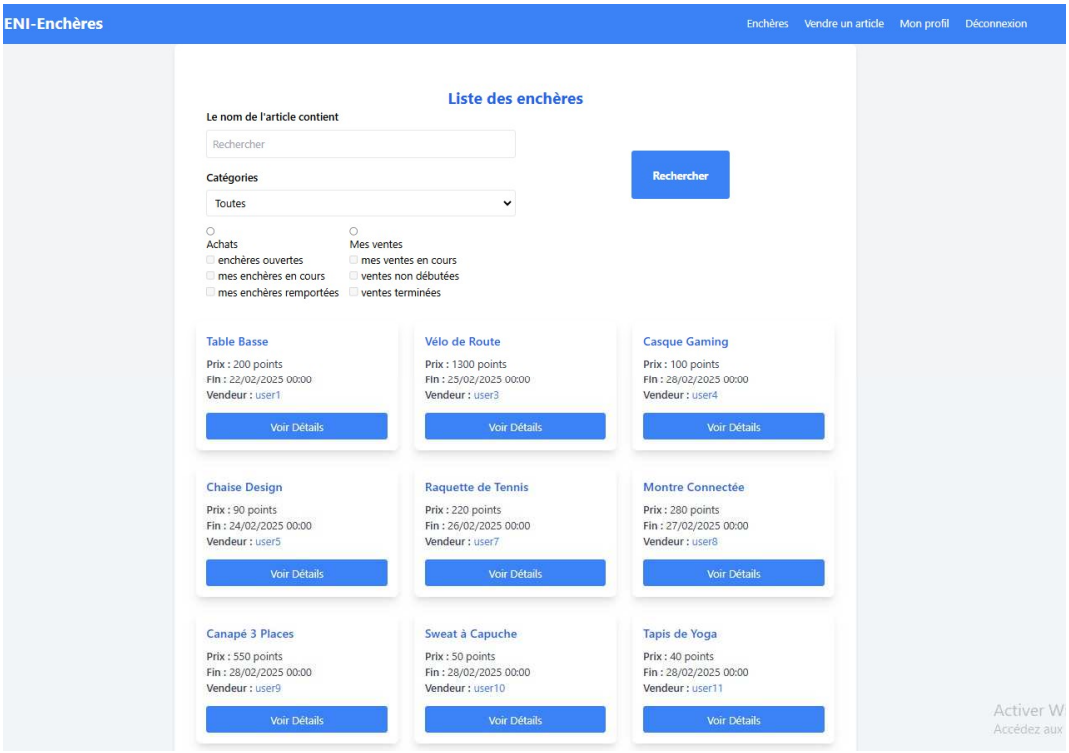
Intitulé
Annexe 1 – Page d'accueil du site d'Enchères
Annexe 2 – Page d'accueil du site d'Enchères – utilisateur connecté
Annexe 3 – Code Javascript pour la gestion dynamique des filtres Achat/Vente dans le formulaire de recherche
Annexe 4 – Code HTML de la page d'accueil, pour les filtres et l'affichage des articles
Annexe 5 – Diagramme de classe
Annexe 6 – Interface ArticleVenduDAO
Annexe 7 – Class ArticleVenduDAOImpl
Annexe 8 – Class EnchereController
Annexe 9 – Interface ArticleVenduService
Annexe 10 – Class ArticleVenduServiceImpl

ANNEXES

Annexe 1 – Page d'accueil du site d'Enchères



Annexe 2 – Page d'accueil du site d'Enchères – utilisateur connecté



Annexe 3 – Code Javascript pour la gestion dynamique des filtres Achat/Vente dans le formulaire de recherche

En fonction du choix les select sont sélectionnables ou non.

```
/**
 * Filtre visuel pour Achat Vente
 */

// Sélection des éléments radio et checkbox
const choix1 = document.getElementById("achats");
const select1 = document.querySelectorAll("#achat1, #achat2, #achat3");
const choix2 = document.getElementById("ventes");
const select2 = document.querySelectorAll("#vente1, #vente2, #vente3");


// Ecoute du click
choix1.addEventListener("click", select);
choix2.addEventListener("click", select);

// Filtre les checkbox selcetionnable en fonction du choix
function select() {
	if (choix1.checked) {
		select2.forEach(checkbox => {
			checkbox.disabled = true;
			checkbox.checked = false;
		})
		select1.forEach(checkbox => {
			checkbox.disabled = false;
		})
	}
	if (choix2.checked) {
		select1.forEach(checkbox => {
			checkbox.disabled = true;
			checkbox.checked = false;
		})
		select2.forEach(checkbox => {
			checkbox.disabled = false;

		})
	}
}
```

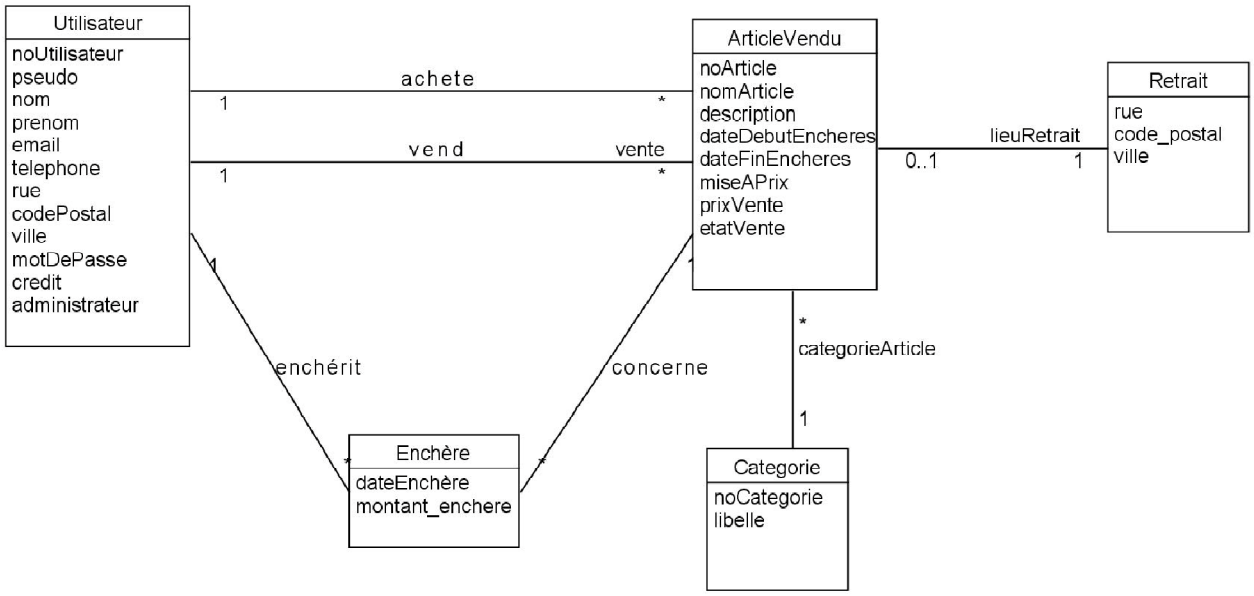
Annexe 4 – Code HTML de la page d'accueil, pour les filtres et l'affichage des articles

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
   <title>Liste des enchères</title>
</head>
<body class="bg-gray-100 p-6">
<header th:replace="~{fragments/fragment-entete :: entete}"></header>

<main class="max-w-5xl mx-auto bg-white shadow-md rounded-lg p-8">
   <h1 class="text-2xl font-bold mt-8 text-blue-600 text-center">Liste des enchères</h1>

   <!-- Formulaire de filtre -->
         <form th:action="@{/encheres/filtre}" method="post" class="mb-6 flex flex-col md:flex-row
md:items-start md:space-x-4">
            <!-- Champ de recherche et Catégorie -->
            <div class="flex flex-col w-full md:w-1/2 space-y-4">
               <div class="w-full">
                  <label for="nom_article" class="block font-semibold">Le nom de l'article contient</label>
                  <input name="motCle" id="nom_article" type="text" placeholder="Rechercher" class="mt-2
p-2 border rounded w-full"/>
               </div>

               <div class="w-full">
                  <label for="categorie" class="block font-semibold">Catégories</label>
                  <select name="categorie" id="categorie" class="mt-2 p-2 border rounded w-full">
                     <option value="0">Toutes</option>
                     <option th:each="c : ${categories}" th:text="${c.libelle}"
th:value="${c.noCategorie}"></option>
                  </select>
               </div>

               <!-- Options de filtre -->
               <div class="flex items-center mt-4" sec:authorize="isAuthenticated()">
                  <div class="flex items-center space-x-4 w-full">
                     <div>
                        <label><input id="achats" type="radio" name="choix" class="block font-
semibold">Achats</label><br>
                        <label><input id="achat1" type="checkbox" name="check" value="achat1" disabled>
enchères ouvertes</label><br>
                        <label><input id="achat2" type="checkbox" name="check" value="achat2" disabled>
mes enchères en cours</label><br>
                        <label><input id="achat3" type="checkbox" name="check" value="achat3" disabled>
mes enchères remportées</label>
```

```
                    </div>
                    <div>
                      <label><input id="ventes" type="radio" name="choix" class="block font-
semibold">Mes ventes</label><br>
                      <label><input id="vente1" type="checkbox" name="check" value="vente1" disabled>
mes ventes en cours</label><br>
                      <label><input id="vente2" type="checkbox" name="check" value="vente2" disabled>
ventes non débutées</label><br>
                      <label><input id="vente3" type="checkbox" name="check" value="vente3" disabled>
ventes terminées</label>
                    </div>
                  </div>
                </div>
              </div>

              <!-- Bouton de recherche centré -->
              <div class="flex items-center justify-center w-full md:w-1/2 mt-4 md:mt-16">
                <button type="submit" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-6 px-8
rounded">Rechercher</button>
              </div>
            </form>


    <!-- Liste des enchères sous forme de vignettes -->
    <div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6 mt-8">
      <div th:each="av : ${article_vendu}" class="bg-white p-4 rounded-lg shadow-lg hover:shadow-xl
transition duration-300">
        <h2 class="text-lg font-semibold text-blue-600">
          <a th:if="${#authorization.expression('isAuthenticated()')}"
            th:href="@{'/articleVendu/detail/' + ${av.noArticle}}"
            th:text="${av.nomArticle}"
            class="hover:underline"></a>
          <span th:unless="${#authorization.expression('isAuthenticated()')}"
th:text="${av.nomArticle}"></span>
        </h2>

        <p class="text-gray-700 mt-2"><span class="font-semibold">Prix :</span><span
th:text="${av.miseAPrix}" th:if="${av.enchere[0].montantEnchere} == 0"></span> <span
th:text="${av.enchere[0].montantEnchere}" th:if="${av.enchere[0].montantEnchere} != 0"></span>
points</p>
        <p class="text-gray-700"><span class="font-semibold">Fin :</span> <span
th:text="${#temporals.format(av.dateFinEncheres, 'dd/MM/yyyy HH:mm')}"></span></p>
        <p class="text-gray-700">
          <span class="font-semibold">Vendeur :</span>
          <a th:if="${#authorization.expression('isAuthenticated()')}"
            th:href="@{'/utilisateurs/detail/' + ${av.vend.noUtilisateur}}"
            th:text="${av.vend.pseudo}"
            class="text-blue-500 hover:underline"></a>
```

```
            <span th:unless="${#authorization.expression('isAuthenticated()')}"
th:text="${av.vend.pseudo}"></span>
          </p>
          <div class="mt-4">
            <a th:if="${#authorization.expression('isAuthenticated()')}"
              th:href="@{'/articleVendu/detail/' + ${av.noArticle}}"
              class="bg-blue-500 hover:bg-blue-700 text-white py-2 px-4 rounded inline-block text-center w-
full">
               Voir Détails
            </a>
          </div>
        </div>
    </div>
</main>

<script src="/js/script-filtre.js"></script>

</body>
</html>
```

Annexe 5 – Diagramme de classe



Annexe 6 – Interface ArticleVenduDAO

```
public interface ArticleVenduDAO {

        List<ArticleVendu> findAll(int noCategorie, String nomArticle, int no_utilisateur, List<String>
check);
}
```

Annexe 7 – Class ArticleVenduDAOImpl

```
@Repository
public class ArticleVenduDAOImpl implements ArticleVenduDAO {

	private final String FIND_ALL = "SELECT av.no_article, av.nom_article, av.description,
av.date_debut_encheres, av.date_fin_encheres, av.prix_initial, av.prix_vente , av.no_utilisateur,
av.no_categorie, u.pseudo, u.no_utilisateur AS u_no_utilisateur, e.no_utilisateur AS u2_no_utilisateur,
e.montant_enchere FROM ARTICLES_VENDUS AS av"
				+ " JOIN UTILISATEURS AS u ON av.no_utilisateur = u.no_utilisateur JOIN
CATEGORIES AS c ON av.no_categorie = c.no_categorie LEFT JOIN ENCHERES e ON e.no_article =
av.no_article AND e.montant_enchere = (SELECT MAX(e2.montant_enchere) FROM ENCHERES e2 WHERE
e2.no_article = av.no_article) WHERE 1=1 ";


	private final String FIND_ARTICLE_BY_CATEGORIE = "AND av.no_categorie = :no_categorie";

	private final String FIND_ARTICLE_BY_NAME = " AND av.nom_article LIKE :nom_article";

	private final String DATE_ENCHERE_EN_COURS = " AND av.date_debut_encheres < GETDATE()
AND av.date_fin_encheres > GETDATE()";

	private final String DATE_FIN_ENCHERE = " AND av.date_fin_encheres < GETDATE()";

	private final String ENCHERE_UTILISATEUR = " AND e.no_utilisateur = :no_utilisateur";

	private final String MES_VENTES = " AND u.no_utilisateur = :no_utilisateur";

	private final String DATE_ENCHERE_NON_DEBUTEE = " AND av.date_debut_encheres >
GETDATE()";

	private final String UPDATE_PRIX_DE_VENTE = "UPDATE ARTICLES_VENDUS SET prix_vente =
:prixVente WHERE no_article = :noArticle";



	@Autowired
	private NamedParameterJdbcTemplate jdbcTemplate;

	/*
	 * Utilisation de la requete WHERE 1=1
	 * teste les conditions des requetes pour chaque filtre dans une seule methode
	 *
	 */
```

```
	@Override
	public List<ArticleVendu> findAll(int noCategorie, String nomArticle, int noUtilisateur, List<String>
	check) {
		MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource();

		String requeteSql = FIND_ALL;

		if (noCategorie != 0) {
			requeteSql += FIND_ARTICLE_BY_CATEGORIE;
			mapSqlParameterSource.addValue("no_categorie", noCategorie);
		}
		if (nomArticle != null && !nomArticle.isBlank() ) {
			requeteSql += FIND_ARTICLE_BY_NAME;
			mapSqlParameterSource.addValue("nom_article", "%" + nomArticle + "%");
		}
		for (String filtreChoix : check) {
			if (filtreChoix.equals("achat1")) {
				requeteSql += DATE_ENCHERE_EN_COURS;
			}
			if (filtreChoix.equals("achat2")) {
				requeteSql += DATE_ENCHERE_EN_COURS + ENCHERE_UTILISATEUR;
				mapSqlParameterSource.addValue("no_utilisateur", noUtilisateur);
			}
			if (filtreChoix.equals("achat3")) {
				requeteSql += DATE_FIN_ENCHERE + ENCHERE_UTILISATEUR;
				mapSqlParameterSource.addValue("no_utilisateur", noUtilisateur);
			}
			if (filtreChoix.equals("vente1")) {
				requeteSql += DATE_ENCHERE_EN_COURS + MES_VENTES;
				mapSqlParameterSource.addValue("no_utilisateur", noUtilisateur);
			}
			if (filtreChoix.equals("vente2")) {
				requeteSql += DATE_ENCHERE_NON_DEBUTEE + MES_VENTES;
				mapSqlParameterSource.addValue("no_utilisateur", noUtilisateur);
			}
			if (filtreChoix.equals("vente3")) {
				requeteSql += DATE_FIN_ENCHERE + MES_VENTES;
				mapSqlParameterSource.addValue("no_utilisateur", noUtilisateur);
			}
		}
		return jdbcTemplate.query(requeteSql, mapSqlParameterSource, new
	 ArticleVenduRowMapper());
	}
}
```

Annexe 8 – Class EnchereController

```
@SessionAttributes({ "utilisateurEnSession", "categorieSession" })
@Controller
public class EncheresController {

	private ArticleVenduService articleVenduService;
	private CategorieService categorieService;

	public EncheresController(CategorieService categorieService, ArticleVenduService
articleVenduService) {
		this.categorieService = categorieService;
		this.articleVenduService = articleVenduService;
	}

	/*
	 * Cette méthode est prise en compte si aucun Utilisateur est connecté
	 * Permet d'envoyer sur la page d'acceuil
	 * par défaut la liste des enchères en cours
	 * On peut filtrer par mot clé et/ou par catégorie
	 * avec le nomArticle, le prix en cours,
	 * la date de fin d'enchère et le nom du vendeur
	 */
	@GetMapping({"/encheres", "/"})
	public String afficherEncheres(@RequestParam(name="motCle", required = false) String
nomArticle, @RequestParam(name="categorie", required = false) String idCategorie,
			@ModelAttribute("utilisateurEnSession") Utilisateur utilisateur, Model model) {

		List<Categorie> categories = categorieService.findAll();
		model.addAttribute("categories", categories);
		int idCategorieInt = 0;
		if (idCategorie != null && !idCategorie.isBlank()) {
			idCategorieInt = Integer.parseInt(idCategorie);
		}
		int noUtilisateur = 0;
		if (utilisateur != null) {
			noUtilisateur = utilisateur.getNoUtilisateur();
		}
		List<ArticleVendu> listeArticleVendu =
this.articleVenduService.consulterLesArticles(nomArticle, idCategorieInt, noUtilisateur);
		model.addAttribute("article_vendu", listeArticleVendu);
		return "index";
	}

	/*
	 * Cette méthode est prise en compte si un Utilisateur est connecté
	 * Permet d'envoyer sur la page d'acceuil
```

```
	 * par défaut la liste des enchères en cours
	 * On peut filtrer par mot clé et/ou par catégorie
	 * et/ou avec les checkbox Achat ou Vente
	 * avec le nomArticle, le prix en cours,
	 * la date de fin d'enchère et le nom du vendeur
	 */

	@PostMapping("/encheres/filtre")
	public String filtreEncheresCategorie(@RequestParam("categorie") String idCategorie,
				@RequestParam("motCle") String nomArticle,
@ModelAttribute("utilisateurEnSession") Utilisateur utilisateur,
				@RequestParam(name="check", required = false) List<String> check, Model
model) {

		List<Categorie> categories = categorieService.findAll();
		model.addAttribute("categories", categories);

		int idCategorieInt = 0;

		if (idCategorie != null && !idCategorie.isBlank()) {
			idCategorieInt = Integer.parseInt(idCategorie);
		}
		int noUtilisateur = 0;

		if (utilisateur != null) {
			noUtilisateur = utilisateur.getNoUtilisateur();
		}
		if (check == null) {
			List<ArticleVendu> filtreListArticleVendu = this.articleVenduService
					.consulterLesArticles(nomArticle, idCategorieInt, noUtilisateur);
			model.addAttribute("article_vendu", filtreListArticleVendu);
		}
		if (check != null) {
			List<ArticleVendu> filtreListAchatVente = this.articleVenduService
					.flitrerLesArticles(idCategorieInt, nomArticle, noUtilisateur, check);
			model.addAttribute("article_vendu", filtreListAchatVente);
		}
		return "index";
	}
}
```

Annexe 9 – Interface ArticleVenduService

```
public interface ArticleVenduService {

	List<ArticleVendu> flitrerLesArticles(int idCategorie, String nomArticle, int noUtilisateur,
List<String> check);
}
```

Annexe 10 – Class ArticleVenduServiceImpl

```
/**
 * Implémentation du service pour la gestion des articles vendus.
 */
@Service
public class ArticleVenduServiceImpl implements ArticleVenduService {


	ArticleVenduDAO articleVenduDAO;
	CategorieDAO categorieDAO;
	RetraitDAO retraitDAO;

	public ArticleVenduServiceImpl(ArticleVenduDAO articleVenduDAO, CategorieDAO categorieDAO,
RetraitDAO retraitDAO) {
		this.articleVenduDAO = articleVenduDAO;
		this.categorieDAO = categorieDAO;
		this.retraitDAO = retraitDAO;

	}

	 /**
   * Consulte les articles vendus en fonction de critères spécifiques.
   *
   * @param nomArticle le nom de l'article à rechercher.
   * @param idCategorie l'identifiant de la catégorie de l'article.
   * @param noUtilisateur le numéro de l'utilisateur associé à l'article.
   * @return une liste d'articles vendus correspondant aux critères.
   */

	@Override
	public List<ArticleVendu> consulterLesArticles(String nomArticle, int idCategorie, int noUtilisateur)
	{
		List<String> listeAchat = new ArrayList<String>();
		listeAchat.add("achat1");
		return this.articleVenduDAO.findAll(idCategorie, nomArticle, noUtilisateur, listeAchat);
	}


	 /**
   * Filtre les articles vendus en fonction de critères spécifiques.
   *
   * @param idCategorie l'identifiant de la catégorie de l'article.
   * @param nomArticle le nom de l'article à rechercher.
   * @param noUtilisateur le numéro de l'utilisateur associé à l'article.
   * @param check une liste de critères supplémentaires pour filtrer les articles.
```

```
    * @return une liste d'articles vendus correspondant aux critères.
    */
        @Override
        public List<ArticleVendu> flitrerLesArticles(int idCategorie, String nomArticle, int noUtilisateur,
List<String> check) {
                return this.articleVenduDAO.findAll(idCategorie, nomArticle, noUtilisateur, check)
        }
}
```